Efficient Decoding Algorithms for Generalised Reed-Muller Codes

Kenneth G. Paterson, Alan E. Jones
Extended Enterprise Laboratory
HP Laboratories Bristol
HPL-98-195
November, 1998

OFDM, power, Reed-Muller code, decoding, algorithms, error correction

Recently a class of generalised Reed-Muller codes has been suggested for use in power-controlled OFDM modulation. A number of approaches to decoding these codes have already been developed. Here we present low complexity alternatives which are inspired by the classical Reed decoding algorithm for binary Reed-Muller codes. We evaluate the decoding performance of these algorithms under realistic channel conditions. We also simulate existing decoding algorithms. We show that one of our new algorithms offers close to maximum likelihood performance and has substantially lower complexity than existing approaches.

Internal Accession Date Only

© Copyright Hewlett-Packard Company 1998

1 Introduction

1.1 Power Controlled Coding for OFDM

A powerful class of codes for Orthogonal Frequency Division Multiplexing (OFDM) have been described in [2, 9]. These codes are constructed from certain generalisations of the classical Reed-Muller codes. As a consequence of this structure, they simultaneously benefit from low peak-to-mean power ratios and have efficient encoding algorithms and good error-correcting capability. This combination of power control and error correction makes the codes extremely attractive for use in portable, low-cost wireless applications of OFDM.

As a means of establishing notation, we give a very brief description of the operation of an OFDM system. An OFDM signal is comprised by adding together n modulated carrier signals with frequencies $f_0 + jf_s, (0 \leq j < n)$:

$$e^{-2\pi i(f_0+jf_s)t} \quad (0 \leq j < n)$$

Under q-PSK modulation, the modulation applied to each carrier is a phase-shift of some integer multiple of $2\pi/q$. Writing $\omega = e^{2\pi i/q}$, the transmitted OFDM signal for the codeword $c = [c_0 c_1 \ldots c_{n-1}]$ (where $c_j \in \mathbb{Z}_q$) is the real part of the sum of phase-shifted carriers:

$$S(c)(t) = \sum_{j=0}^{n-1} \omega^{c_j} e^{-2\pi i(f_0+jf_s)t}.$$

Typically, q is equal to 2,4 or 8 and n is a power of 2 (this allows the use of efficient FFT-based signal processing techniques). Other modulation schemes, notably QAM, have also been proposed for OFDM, but we consider only constant-amplitude modulation schemes in this paper. At the receiver, a noisy version of the signal $\mathrm{Re}(S(c)(t))$ is received and sampled. The samples are preprocessed to perform synchronisation, decimation and guard period removal. Then demodulation is performed using a Fourier transform and a multipath-faded and noise-corrupted version r of the vector $\omega^c = [\omega^{c_0}, \omega^{c_1}, \ldots \omega^{c_{n-1}}]$ is recovered. Typically, it is assumed that the noise on each component of r is Gaussian distributed. We refer to Section 6 for a more detailed description of the multipath channel model that we have used in our simulations.

A key contribution of [2, 9] is to identify classes of codewords c for which the function $|S(c)(t)|^2$, called the instantaneous envelope power of the signal, is limited in maximum value. This feature is essential for the practical use of OFDM in applications where electronic components need to be of low cost or where tight control of the spectral power of the transmitted signals is required [6]. The OFDM codes of [2, 9] are obtained as unions of cosets of a q-ary generalisation of the first-order Reed-Muller code, denoted $RM_q(1, m)$, lying inside the generalised second-order codes $RM_q(2, m)$ and $ZRM_q(2, m)$ (these families of codes will be defined in Section 2.1 below). Work in [8, 11] also identified a limited subset of these codes, but without making explicit the link to Reed-Muller codes.

To exploit the error-correcting capability of these OFDM codes in low-cost wireless applications, low complexity, high performance decoding algorithms are needed. Already several approaches to this problem have been developed [2, 3, 4, 5, 11]. But, in many situations of practical interest, these algorithms either have prohibitive complexity or do not realise the full potential of the generalised Reed-Muller codes. The problem is particularly acute for the high rate codes of [9] which are formed from moderate to large numbers of cosets of the Reed-Muller codes.

1.2 Our Contribution

This paper makes three main contributions to the decoding of generalised Reed-Muller codes for OFDM.

Firstly, we present two new theoretical insights. The first insight enables us to develop analogues of the classical Reed majority logic decoding algorithm [7, pp. 385–388] that are suitable for generalised Reed-Muller codes. These algorithms significantly advance on decoding approaches contained in [11]. The second insight is an algorithmic step which efficiently reduces the decoding of the r-th order code $RM_q(r, m)$ to m decodings of the $(r-1)$-st order code $RM_q(r-1, m-1)$. This step appears to be new even for the classical binary codes. Applied to the second-order code, it allows us to make repeated use of any first-order algorithm we please. We prove theorems evaluating the decoding performance of the various algorithms, in terms of the maximum weight of a correctable error.

Secondly, we develop two low complexity, soft-decision decoding algorithms that are specifically applicable to OFDM codes formed from moderate to large numbers of cosets of the first-order Reed-Muller codes of the type presented in [2, 9]. Algorithm 6 uses majority logic decoding, while Algorithm 8 makes use of the new reduction step and of the q-ary generalisation of the Fast Hadamard Transform (FHT) given in [4].

Thirdly, we make a comparison of our new algorithms with the previous approaches to decoding generalised Reed-Muller codes given in [2, 3, 4, 5, 11]. We evaluate the complexities of the various algorithms and simulate their performance in Gaussian noise and with a multipath channel. This gives the first direct comparison of all the competing decoding strategies. For a particular code and channel, we show that our Algorithm 8 out-performs every other algorithm with the exception of the computationally intensive maximum likelihood approach, and comes within 2dB of the performance of that algorithm. Algorithm 8 is also significantly more efficient than previously proposed algorithms, both for this example code and many other codes of practical interest.

1.3 Paper Organisation

In Section 2 we provide necessary background on generalised Reed-Muller codes and the previous approaches to decoding. Then in Section 3 we give a generalisation of the classical Reed decoding algorithm for the code $RM(1, m)$ to the q-ary case. Like its binary fore-bearer, the algorithm has rather poor performance. However, it serves as a useful introduction to the ideas behind our decoders for the full second-order code $RM_q(2, m)$ and codes formed from unions of cosets of $RM_q(1, m)$ inside $RM_q(2, m)$ that we present in Section 4.

Our second strand of algorithms come from a new algorithmic step which reduces the decoding of $RM_q(r, m)$ to m decodings of the code $RM_q(r-1, m-1)$. In Section 5, we describe this step and the resulting decoders for the full second-order code and for codes formed from unions of cosets of $RM_q(1, m)$ inside $RM_q(2, m)$.

In Section 6 we compare the implementation complexities of the various decoding strategies, old and new, with particular reference to an OFDM code that is formed from 32 cosets of $RM_4(1, 4)$. We also report the results of simulations for the algorithms, with reference to the same OFDM code.

2 Coding Background

2.1 Generalised Reed-Muller Codes

Throughout, q denotes an even integer, and r and m are integers with $0 \leq r \leq m$. We define a *generalised Boolean function* to be a mapping $f : \{0,1\}^m \to \mathbb{Z}_q$ of $\{0,1\}$-valued variables $x_0, x_1, \ldots, x_{m-1}$. A straightforward counting argument shows that every such function can be written in algebraic normal form as a sum of monomials of the form $x_{j_0} x_{j_1} \ldots x_{j_{r-1}}$ (in which $j_0, j_1 \ldots, j_{r-1}$ are distinct). With each generalised Boolean function f we identify a length 2^m $\mathbb{Z}_q$-valued vector $[f_0 f_1 \ldots f_{2^m-1}]$ in which

$$f_i = f(i_0, i_1, \ldots, i_{m-1})$$

where $[i_0 i_1 \ldots i_{m-1}]$ is the binary expansion of the integer i (so that $i = \sum_{j=0}^{m-1} i_j 2^j$). Henceforth we identify generalised Boolean functions and their corresponding vectors, using f to refer to both.

We recall the definitions of the codes $RM_q(r,m)$ and $ZRM_q(r,m)$ from [2, 9]: $RM_q(r,m)$ is the length 2^m linear code over $\mathbb{Z}_q$ that is generated by the monomials of order at most r in variables $x_0, \ldots, x_{m-1}$, while for q even, $ZRM_q(r,m)$ is the length 2^m linear code over $\mathbb{Z}_q$ that is generated by the monomials of order at most $r-1$ together with twice the monomials of order r in $x_0, \ldots, x_{m-1}$.

Alternatively, $RM_q(r,m)$ is the q-ary code whose codewords are obtained as all the $\mathbb{Z}_q$-linear combinations of the rows of the generator matrix for the classical binary Reed-Muller code $RM(r,m)$.

Example 1 *The code $RM_4(2,4)$ is the linear code over $\mathbb{Z}_4$ with generator matrix:*

$$\begin{bmatrix} 1111111111111111 \\ 0101010101010101 \\ 0011001100110011 \\ 0000111100001111 \\ 0000000011111111 \\ 0001000100010001 \\ 0000010100000101 \\ 0000001100000011 \\ 0000000001010101 \\ 0000000000110011 \\ 0000000000001111 \end{bmatrix} \begin{matrix} 1 \\ x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_0x_1 \\ x_0x_2 \\ x_1x_2 \\ x_0x_3 \\ x_1x_3 \\ x_2x_3 \end{matrix}$$

while $ZRM_4(2,4)$ is the linear code over $\mathbb{Z}_4$ with generator matrix:

$$\begin{bmatrix} 1111111111111111 \\ 0101010101010101 \\ 0011001100110011 \\ 0000111100001111 \\ 0000000011111111 \\ 0002000200020002 \\ 0000020200000202 \\ 0000002200000022 \\ 0000000002020202 \\ 0000000000220022 \\ 0000000000002222 \end{bmatrix} \begin{matrix} 1 \\ x_0 \\ x_1 \\ x_2 \\ x_3 \\ 2x_0x_1 \\ 2x_0x_2 \\ 2x_1x_2 \\ 2x_0x_3 \\ 2x_1x_3 \\ 2x_2x_3 \end{matrix}$$

Typical codewords of these codes are:

$$\begin{aligned} f &= 1 + x_0 + 3x_1 + 2x_3 = [1201120130233023] \in RM_4(1,4), \\ f' &= 1 + x_0 + 3x_1 + 2x_0x_1 + 2x_0x_2 + 2x_2x_3 = [1203100112033223] \in ZRM_4(2,4). \end{aligned}$$

By a coset of the code $RM_q(1,m)$ we mean a set of the form $f + RM_q(1,m)$ where f is a q-ary length 2^m word and '+' denotes componentwise addition of vectors modulo q. We call f a coset representative. Any coset of $RM_q(1,m)$ in $RM_q(2,m)$ can be identified with a coset representative Q that is a quadratic form in variables $x_0, x_1, \ldots, x_{m-1}$:

$$Q = \sum_{0 \le j < k < m} q_{jk} x_j x_k$$

where $q_{ij} \in \mathbb{Z}_q$. When all the coefficients q_{ij} are even, the coset $Q + RM_q(1,m)$ lies in $ZRM_q(2,m)$.

We next give an example of the OFDM codes that were presented in [2, 9]. The code is typical in that it consists of a moderate number of cosets of a short first-order code.

Example 2 *We use a code of the above type to compare the performance of the various decoding algorithms in Section 6. The code consists of 32 cosets of the quaternary code $RM_4(2,4)$ inside $ZRM_4(2,m)$. Thus the code contains $32 \cdot 4^5 = 2^{15}$ codewords and can be used to encode 15 information bits (so the rate is $15/32 = 0.47$). Because it is contained in $ZRM_4(2,4)$, the code has minimum Hamming distance at least 4 and minimum Lee distance at least 8 (see Theorem 1). The 32 coset representatives for the code are:*

$2x_0x_1 + 2x_2x_3$ $\quad$ $2x_0x_2 + 2x_1x_3$
$2x_0x_3 + 2x_1x_2$ $\quad$ $2x_0x_1 + 2x_0x_2 + 2x_1x_3$
$2x_0x_1 + 2x_0x_2 + 2x_2x_3$ $\quad$ $2x_0x_1 + 2x_0x_3 + 2x_2x_3$
$2x_0x_1 + 2x_1x_2 + 2x_2x_3$ $\quad$ $2x_0x_1 + 2x_1x_2 + 2x_0x_3$

$2x_0x_1 + 2x_1x_3 + 2x_2x_3$ $\quad$ $2x_0x_2 + 2x_0x_3 + 2x_1x_2$
$2x_0x_2 + 2x_0x_3 + 2x_1x_3$ $\quad$ $2x_0x_2 + 2x_1x_2 + 2x_1x_3$
$2x_0x_2 + 2x_1x_3 + 2x_2x_3$ $\quad$ $2x_0x_3 + 2x_1x_2 + 2x_1x_3$
$2x_0x_3 + 2x_1x_2 + 2x_2x_3$ $\quad$ $2x_0x_1 + 2x_0x_2 + 2x_0x_3 + 2x_1x_2$

$2x_0x_1 + 2x_0x_2 + 2x_0x_3 + 2x_1x_3$ $\quad$ $2x_0x_1 + 2x_0x_2 + 2x_0x_3 + 2x_2x_3$
$2x_0x_1 + 2x_0x_2 + 2x_1x_2 + 2x_1x_3$ $\quad$ $2x_0x_1 + 2x_0x_2 + 2x_1x_2 + 2x_2x_3$
$2x_0x_1 + 2x_0x_2 + 2x_1x_3 + 2x_2x_3$ $\quad$ $2x_0x_1 + 2x_0x_3 + 2x_1x_2 + 2x_1x_3$
$2x_0x_1 + 2x_0x_3 + 2x_1x_2 + 2x_2x_3$ $\quad$ $2x_0x_1 + 2x_0x_3 + 2x_1x_3 + 2x_2x_3$

$2x_0x_1 + 2x_1x_2 + 2x_1x_3 + 2x_2x_3$ $\quad$ $2x_0x_2 + 2x_0x_3 + 2x_1x_2 + 2x_1x_3$
$2x_0x_2 + 2x_0x_3 + 2x_1x_2 + 2x_2x_3$ $\quad$ $2x_0x_2 + 2x_0x_3 + 2x_1x_3 + 2x_2x_3$
$2x_0x_2 + 2x_1x_2 + 2x_1x_3 + 2x_2x_3$ $\quad$ $2x_0x_3 + 2x_1x_2 + 2x_1x_3 + 2x_2x_3$
$2x_0x_1 + 2x_0x_2 + 2x_0x_3 + 2x_1x_2 + 2x_2x_3$ $\quad$ $2x_0x_1 + 2x_0x_2 + 2x_1x_2 + 2x_1x_3 + 2x_2x_3$

The signals $S(c)(t)$ corresponding to codewords from this code all have peak envelope power at most 64, so the peak-to-mean envelope power (PMEPR) of the code is at most 4. This should be compared to a PMEPR of 16 for an uncoded OFDM system with the same number of carriers.

2.2 Lee and Hamming Metrics

Let $x = [x_0 x_1 \ldots x_n]$ and $y = [y_0 y_1 \ldots y_{n-1}]$ be $\mathbb{Z}_q$-valued vectors of length n.

We define the *Hamming weight* of x, denoted $wt_H(x)$ to be the real sum $\sum_{x_i \neq 0} 1$ which counts the number of non-zero components in x. We define the *Hamming distance* between x and y, denoted $d_H(x, y)$ to be $wt_H(x - y)$ (where subtraction is componentwise modulo q). The minimum Hamming distance of a code $\mathcal{C}$ over $\mathbb{Z}_q$ is defined to be $\min_{x,y \in \mathcal{C}} d_H(x, y)$ and is denoted by $d_H(\mathcal{C})$.

We define the *Lee weight* of x, denoted $wt_L(x)$ to be the real sum $\sum_{i=0}^{n-1} \min\{x_i, q - x_i\}$. We define the *Lee distance* between x and y, denoted $d_L(x, y)$ to be $wt_L(x - y)$ (where subtraction is componentwise modulo q). The minimum Lee distance of a code $\mathcal{C}$ over $\mathbb{Z}_q$ is defined to be $\min_{x,y \in \mathcal{C}} d_L(x, y)$ and is denoted by $d_L(\mathcal{C})$.

We have the following theorem from [2, 9] concerning minimum Hamming and Lee distances of the codes $RM_q(r, m)$ and $ZRM_q(r, m)$:

Theorem 1 *We have:*

- *for* $q \geq 2$, $d_H(RM_q(r, m)) = d_L(RM_q(r, m)) = 2^{m-r}$.
- *for* $q \geq 4$ *with* q *even,* $d_H(ZRM_q(1, m)) = 2^{m-r}$ *and* $d_L(ZRM_q(1, m)) = 2^{m-r+1}$.

We also require a notion of Lee weights and distances that is applicable to real-valued vectors $x = [x_0 x_1 \ldots x_{n-1}]$ and $y = [y_0 y_1 \ldots y_{n-1}]$ with $x_i, y_i \in [0, q)$. We define the *soft Lee weight* of x, again denoted $wt_L(x)$, to be the real sum:

$$\sum_{i=0}^{n-1} \min\{x_i, q - x_i\}$$

and the *soft Lee distance* between x and y, again denoted $d_L(x, y)$, to be $wt_L((x - y) \bmod q)$ where $(x - y) \bmod q$ denotes the vector whose components are the unique real numbers z_i in the range $[0, q)$ satisfying $z_i - (x_i - y_i) = \ell q$ for some integer ℓ.

2.3 Existing Decoding Approaches

In the binary case, $q = 2$, the generalised Reed-Muller codes coincide with the classical Reed-Muller codes. So well established techniques can be used to handle decoding of the binary OFDM codes. For example, a maximum likelihood, soft-decision (MLSD) algorithm can be obtained by combining the method of supercode decoding [1] with the Fast Hadamard Transform (FHT) MLSD decoder for $RM(1, m)$ [7, pp. 419–426].

Supercode decoding is a general method applicable to codes formed from a union of cosets of a base code. Each coset representative is subtracted from the received word in turn, and the best result (in some metric sense) obtained using a decoder for the base code over all these modified words is selected. The corresponding coset representative and this best result determine the final decoded word. The FHT algorithm gives a computationally efficient method for computing the correlations between a received word and *all* 2^{m+1} words of the code $RM(1, m)$.

For a length 2^m code formed from ℓ cosets, the complexity of a soft-decision 'supercode+FHT' approach is approximately $\ell m 2^m$ real operations. So this maximum likelihood supercode approach is computationally feasible only when ℓ is relatively small. A convenient method for handling large numbers of binary cosets is to regard a received word as a codeword of the full second-order code (with the addition of noise) and then use the well-known Reed decoding

algorithm [7, pp. 385–388]. This approach is guaranteed only to correct errors whose weights are less than half the minimum distance of the second-order Reed-Muller code $RM(2, m)$, i.e. whose weights are less than 2^{m-3}. It is not maximum likelihood in general and gives no information when the decoded word happens to lie in a coset of $RM(1, m)$ inside $RM(2, m)$ that is *not* in the original union of cosets. Moreover, the Reed algorithm as originally described is a hard-decision algorithm. In other words, it operates on an input vector of binary-valued components and does not use additional soft information that may be available from the OFDM demodulator.

For more general q-ary Reed-Muller codes, there already exist several decoding algorithms [2, 3, 4, 11]. These can be roughly characterised as being of two types: 'signal-domain' algorithms and 'coding-domain' algorithms. The former type has as inputs complex values that are obtained directly from the vector output by the OFDM demodulator (which we have denoted by r above) while the latter has inputs which are real numbers in the range $[0, q)$, obtained by appropriately scaling the phases of the components of r. Coding-domain algorithms can be further classified into hard-decision and soft-decision algorithms. In the hard-decision case, the phase information is quantised to integer values, so that the input to the decoder is a vector of integers. In soft-decision decoding, the real-number phase information is directly passed to the decoder.

Essentially, a coding-domain algorithm rejects magnitude information from the demodulated signal while a signal-domain algorithm preserves it. A potential advantage of a coding-domain algorithm is that it may need to use only real operations (or integer operations in the hard-decision case) while a signal-domain one will most naturally use complex operations and as such may be more expensive to implement. On the other hand, for multipath fading channels, magnitude information is an important indicator of symbol reliability and a signal-domain algorithm has the potential to use this information to obtain enhanced decoding performance.

A signal-domain version of the Reed majority logic decoding algorithm suited to the codes $RM_q(1, m)$ was given in [11], though it was presented there only for a length 8 example of what van Nee called a 'complementary code'. Nevertheless, this algorithm inspired the algorithms of Section 3 in this paper and indeed is a special case of Algorithm 2. We will see in Section 3 that its complexity is prohibitively large for codes formed from many cosets of $RM_q(1, m)$. The decoding performance for a single coset of the length 8 octary code is estimated [11] as being 3dB worse than that of maximum likelihood decoding.

In [3, 4], a q-ary analogue of the FHT is developed which yields a signal-domain MLSD algorithm for the q-ary generalised first-order code $RM_q(1, m)$. This algorithm computes the correlations between the received word r and the complex versions of all codewords of $RM_q(1, m)$ of the form $\sum_{k=0}^{m-1} a_k x_k$, i.e. all codewords having zero constant coefficient. This computation can be expressed as a matrix multiplication:

$$Y = Hr. \tag{1}$$

Here H is a $q^m \times 2^m$ matrix whose rows, labelled $H_{[a_0 a_1 \ldots a_{m-1}]}$ are the complex-conjugated versions ω^{-c} of codewords $c = \sum_{k=0}^{m-1} a_k x_k$ and Y is complex vector with entry $Y_{[a_0 a_1 \ldots a_{m-1}]}$ being the correlation $\sum_{i=0}^{2^m-1} \omega^{-c_i} \cdot r_i$. The entry of maximum absolute value in Y, say $Y_{[\hat{a}_0 \hat{a}_1 \ldots \hat{a}_{m-1}]}$, determines a codeword of the form $\sum_{k=0}^{m-1} \hat{a}_k x_k$. The maximum likelihood estimate of the transmitted codeword is then $\sum_{k=0}^{m-1} \hat{a}_k x_k + \hat{a}$ where $\hat{a} \in \mathbb{Z}_q$ is the integer which maximises $\mathrm{Re}(\omega^{-\hat{a}} Y_{[\hat{a}_0 \hat{a}_1 \ldots \hat{a}_{m-1}]})$. The output of the decoder is the list of q-ary coefficients $\hat{a}, \hat{a}_0, \hat{a}_1 \ldots, \hat{a}_{m-1}$.

A naive approach to the computation in equation (1) would require on the order of $2^m q^m$ complex arithmetic operations. Substantial reductions in complexity can be obtained by decomposing the matrix multiplication in a similar way as in the binary FHT described in [7, pp.

419–426] — see [4] for details. In particular, for $q = 4$, all multiplications can be avoided and the number of complex additions required is on the order of 2^{2m}. More generally, for $q = 2^h$, this algorithm requires on the order of 2^{hm} complex additions and, for $h \geq 3$, an additional 2^{hm} complex multiplications. It is therefore too complex for all but the shortest codes and infeasible for use in supercode decoding when ℓ is even of moderate size.

A more efficient approach to decoding was developed in [2] for 2^h-ary codes, the case of most practical interest. There, coding-domain decoders for $RM_{2^h}(1, m)$ requiring the computation of only h length 2^m integer or real FHTs (for hard- or soft-decision decoding) were given. Consequently, the complexity of these decoders is on the order of $hm2^m$ operations. The algorithms are not maximum likelihood, but a set of error patterns that they can correct were classified in [2]. In particular, the algorithms are minimum distance decoders for both Hamming and Lee metrics (i.e. they can correct all errors of Hamming or Lee weight less than half the appropriate minimum distance of the first-order code). These algorithms are closely related to the sub-optimal signal-domain algorithm presented in [3, Section V].

Further work in [2] combined the above coding-domain algorithm in a non-trivial way with a generalisation of the supercode method to produce decoders that are applicable to codes formed from unions of ℓ cosets of $RM_{2^h}(1, m)$ which require at most $\ell + h - 1$ real FHTs and so have complexity on the order of $(\ell + h - 1)m2^m$ real operations.

We also mention the decoding algorithm for quaternary codes ($q = 4$) given in [5]. This is a maximum likelihood, hard-decision, coding-domain algorithm which makes neat use of the existence of a distance-preserving Gray map between the length 2^m quaternary code $ZRM_4(1, m)$ and the length 2^{m+1} binary code $RM_2(1, m+1)$. Since $RM_4(1, 4)$ can be represented as a union of 2^m cosets of $ZRM_4(1, m)$, this map sends any union of ℓ cosets of $RM_4(1, m)$ onto a union of $2^m\ell$ cosets of $RM(1, m + 1)$ and allows the use of binary decoding techniques (for example, binary FHTs) to be applied to a quaternary code. By carefully extending the Gray map to soft values, it is also possible to develop a soft-decision version of this algorithm. We include the details for completeness. If we use the standard Gray map

$$\phi : 0 \to (0, 0),\ 1 \to (0, 1),\ 2 \to (1, 1),\ 3 \to (1, 0),$$

then we can extend ϕ to a soft Gray map on inputs $r \in [0, 4)$ by writing:

$$\phi(r) = \begin{cases} (0, r) & \text{for } 0 \leq r < 1, \\ (r - 1, 1) & \text{for } 1 \leq r < 2, \\ (1, 3 - r) & \text{for } 2 \leq r < 3, \\ (4 - r, 0) & \text{for } 3 \leq r < 4. \end{cases}$$

We can apply this soft ϕ to the components of real-valued input vectors and then use real-input FHTs on the resulting length 2^{m+1} vectors. Unfortunately, this decoder requires the computation of $2^m\ell$ FHTs and this makes the decoder too intensive in all but the simplest of instances.

2.4 Further Facts about Generalised Reed-Muller Codes

We prove some simple facts about generalised Reed-Muller codes which will form the basis for our new decoding algorithms in subsequent sections.

Definition 1 *Let k be an integer with $0 \leq k < m$ and let $\mathcal{I}_k$ be the set of integers i with $0 \leq i = \sum_{\alpha=0}^{m-1} i_\alpha 2^\alpha < 2^m$ with the property that $i_k = 0$. Given f, a length 2^m vector over $\mathbb{Z}_q$, we define f^k to be the length 2^{m-1} vector with components f^k_i, $i \in \mathcal{I}_k$ where*

$$f^k_i = f_{i+2^k} - f_i \bmod q, \quad i \in \mathcal{I}_k$$

For example the set $\mathcal{I}_0$ is equal to $\{0, 2, 4, \ldots, 2^m - 2\}$ and consists of all the even integers between 0 and $2^m - 2$. Similarly, $\mathcal{I}_{m-1}$ consists of all integers between 0 and $2^{m-1} - 1$. It is also easy to see that, in general, the set $\mathcal{I}_k$ contains exactly those positions in which the codeword x_k has a zero. If $m = 4$, $q = 4$ and $f = (1211120130233021)$, then

$$\begin{aligned} f^0 &= [10111113], \\ f^1 &= [03333331], \\ f^2 &= [00300002], \\ f^3 &= [22122220]. \end{aligned}$$

Lemma 2 *Let f be a codeword of $RM_q(r, m)$. Then for $0 \leq k < m$, f^k is the codeword of $RM_q(r-1, m-1)$ corresponding to the generalised Boolean function*

$$f(x_0, ..., x_{k-1}, 1, x_{k+1}, \ldots, x_{m-1}) - f(x_0, ..., x_{k-1}, 0, x_{k+1}, \ldots, x_{m-1}) \bmod q,$$

in variables $x_0, \ldots, x_{k-1}, x_{k+1}, \ldots, x_{m-1}$, which is obtained by substituting $x_k = 1$ and $x_k = 0$ into the algebraic normal form for f, subtracting the results modulo q and simplifying.

Proof: Given i with $i \in \mathcal{I}_k$, consider the vector f in positions i and $i+2^k$. The binary expansions of these indices differ in the k-th bit only, where i has a 0 and $i + 2^k$ a 1. Components f_i and f_{i+2^k} of f are therefore obtained by evaluating the generalised Boolean function at inputs which differ only in the value of the variable x_k. It is now clear that the vector f^k corresponds to the Boolean function in the statement of the lemma. The monomials which appear in this function are obtained by

- removing all monomials not involving x_k in f,
- replacing x_k by 1 in all monomials which do involve x_k in f

That the order of the resulting function is at most $r - 1$ is then obvious. □

Example 3 *Take $c = 1 + x_0 + 3x_1 + 2x_3 = [1201120130233023] \in RM_4(1, 4)$,*
$e = [0010000000000002]$ and $r = c + e \bmod 4 = [1211120130233021]$. Then $c^3 = [22222222]$, while

$$c(x_0, x_1, x_2, 1) - c(x_0, x_1, x_2, 0) = 2.$$

Notice that c^3 is a constant word (of $RM_4(0, 3)$) whose components all equal the coefficient of x_3 in c. On the other hand, $r^3 = [22122220]$ and we see that the majority *of the components in r^3 still equal 2, despite the presence of the error vector e.*

The above example reveals the basis for our decoding algorithms: simple transformations of Reed-Muller codewords corrupted by noise still reveal coefficients used in the encoding process.

3 Majority Logic Decoding Algorithms for First-order Codes

We present generalisations of the classical majority logic decoding algorithm that are applicable to the codes $RM_q(1, m)$.

3.1 Majority Logic Coding-Domain Decoders for $\mathbf{RM_q(1,m)}$

Our first decoder is a coding-domain Hamming and Lee distance hard-decision decoder.

Algorithm 1 *(Hard-decision coding-domain decoding algorithm for $RM_q(1,m)$).*

1. *Input the received word $r = [r_0 r_1 \dots r_{2^m-1}]$ as a vector over $\mathbb{Z}_q$.*
2. *For $k = 0, 1, \dots, m-1$, compute the length 2^{m-1} vector r^k (as in Definition 1).*
3. *For $k = 0, 1, \dots, m-1$, find the most frequent symbol in r^k. Denote this symbol by $\hat{a}_k$.*
4. *Let $r' = r - \sum_{k=0}^{m-1} \hat{a}_k x_k \bmod q$.*
5. *Find the most frequent symbol in r'. Denote this symbol by $\hat{a}$.*
6. *Output $\hat{a}, \hat{a}_0, \hat{a}_1, \dots, \hat{a}_{m-1}$*

The relationship between Algorithm 1 and the Reed majority logic algorithm for $RM(1,m)$ is straightforward: modulo 2 addition is replaced with appropriate modulo q operations, and in place of taking a majority decision (between symbols 0 and 1), the most common symbol amongst q possible symbols is selected. The algorithm reduces to the Reed algorithm when $q = 2$.

Theorem 3 *Algorithm 1 is a Hamming and Lee distance hard-decision decoder for the code $RM_q(1,m)$.*

Proof: Let $c = a + \sum_{i=0}^{m-1} a_k x_k$ be the transmitted codeword, and let r be the received word. Write $e = r - c \bmod q$ so that e represents the error vector. Since the minimum Hamming and Lee distances of $RM_q(1,m)$ are 2^{m-1}, we need only show that, under either the restriction $wt_H(e) < 2^{m-2}$ or the restriction $wt_L(e) < 2^{m-2}$, Algorithm 1 computes values $\hat{a}_0, \hat{a}_1, \dots, \hat{a}_{m-1}, \hat{a}$ satisfying:

$$\hat{a}_0 = a_0, \hat{a}_1 = a_1, \dots, \hat{a}_{m-1} = a_{m-1}, \hat{a} = a.$$

Now, for each k with $0 \le k < m$, we have $r^k = c^k + e^k \bmod q$. From Lemma 2, we know that c^k is equal to the vector $[a_k a_k \dots a_k]$. Since e has Hamming or Lee weight less than 2^{m-2}, so does the vector e^k (obtained by subtracting pairs of components of e). So less than half the entries in c^k are changed from a_k to other values when adding e_k. Thus the most common entry in r^k is still a_k, and so the value $\hat{a}_k$ output by the algorithm equals a_k. Continuing, we then find that

$$\begin{aligned} r' &= r - \sum_{k=0}^{m-1} \hat{a}_k x_k \\ &= e + a + \sum_{k=0}^{m-1} (a_k - \hat{a}_k) x_k \\ &= e + a \end{aligned}$$

Again, because of the restricted weight of e, we see that the majority of the components of r' are equal to a. Hence the algorithm outputs a value $\hat{a}$ that is equal to a. □

The computational cost of the above algorithm is made up from $(m+2)2^{m-1}$ subtractions modulo q (step 2), $(m+2)2^{m-1}$ comparisons (steps 3 and 5) and the cost of one encoding step

and 2^m subtractions modulo q (step 4). Ignoring the cost of re-encoding (which can be achieved by a simple hardware circuit), the total cost is about that of $(m+3)2^m$ additions modulo q. This is slightly greater than for the FHT-based decoder applicable to the binary code $RM(1,m)$ (which requires on the order of $m2^m$ additions modulo 2 for hard-decision decoding), but is always less (and often, substantially less) than the decoders of [2, Algorithm 5.3], [3, Section V] and [4, Algorithm 1] when $q \geq 4$. Of course, we expect these latter algorithms to have superior decoding performance.

We note that steps 3 and 5 of Algorithm 1 are really hard-decision decoding algorithms for the codes $RM_q(0,m-1)$ and $RM_q(0,m)$: this code has as codewords the constant vectors $[aa\ldots a]$ and maximum likelihood hard-decision decoding of these codes is achieved simply by finding the most common symbol in the received word.

We can also derive a less efficient, soft-decision decoder for $RM_q(1,m)$ using similar ideas. For soft-decision decoding, each component of the received vector r in step 1 is a real number in the range $[0,q)$ and in the computation of r^k in step 2, we take $r_i - r_{i+2^k} \mod q$ to be the unique real number x in the range $[0,q)$ satisfying $x - (r_i - r_{i+2^k}) = \ell q$ for some integer ℓ. Similarly in step 4, the computation of r' is carried out using real arithmetic operations and each component is forced to lie in the range $[0,q)$. Finally, in steps 3 and 5, we cannot simply make a frequency-based decision for the values of $\hat{a}_k$ and a, because the symbols in the vectors r^k and r' are no longer from the alphabet $\mathbb{Z}_q$. An alternative step is to choose for $\hat{a}_k$ the value which minimises the soft Lee distance $d_L(\hat{a}_k, r^k)$. Here, $\hat{a}_k$ denotes a constant vector of length 2^{m-1}. Similarly for the coefficient $\hat{a}$. The computational cost of this for each coefficient $\hat{a}_k$ is roughly $q2^m$ real operations. The total computational requirement is then around $(2q+1)(m+2)2^{m-1}$ real arithmetic operations (ignoring the cost of re-encoding in Step 4). It is easily shown that this procedure can correct all errors e whose soft Lee weight is less than 2^{m-2}. We will give an alternative signal-domain soft-decision algorithm in the next sub-section.

3.2 Majority Logic Signal-Domain Decoder for $\mathbf{RM_q(1,m)}$

Now we give a signal-domain decoder for $RM_q(1,m)$. This decoder is similar to the one presented in [11, Section V] for the length 8 'complementary code'.

Algorithm 2 *(Signal-domain decoding algorithm for $RM_q(1,m)$).*

1. *Input the received word $r = [r_0 r_1 \ldots r_{2^m-1}]$ as a vector over $\mathbb{C}$.*
2. *For $k = 0, 1, \ldots, m-1$, compute the complex number*
$$R_k = \sum_{i \in \mathcal{I}_k} r_{i+2^k} \cdot r_i{}^*.$$
3. *For $k = 0, 1, \ldots, m-1$, select from $\mathbb{Z}_q$ the value $\hat{a}_k$ which maximises $Re(\omega^{-\hat{a}_k} R_k)$.*
4. *Compute the codeword $z = \sum_{k=0}^{m-1} \hat{a}_k x_k \mod q$ and the complex number*
$$R' = \sum_{i=0}^{2^m-1} r_i \cdot \omega^{-z_i}.$$
5. *Select from $\mathbb{Z}_q$ the value $\hat{a}$ which maximises $Re(\omega^{-\hat{a}} R')$.*

6. *Output* $\hat{a}, \hat{a}_0, \hat{a}_1, \ldots, \hat{a}_{m-1}$

To obtain Algorithm 2 from Algorithm 1, we have essentially replaced modulo q operations by appropriate signal-domain analogues at each step. We have also replaced the strategy of making a majority decision for the coefficients $\hat{a}_k$ and $\hat{a}$ by a step appropriate to the complex domain of 'averaging the phase' in the complex versions of the vectors r^k and r' (steps 2 and 4). Steps 2 and 4 are effectively signal-domain decoders for $RM_q(0, m-1)$ and $RM_q(0, m)$ respectively. In fact, it is not hard to show that if $R = \sum_{i=0}^{2^m-1} r_i$ where $r = [r_0 r_1 \ldots r_{2^m-1}]$ is the received complex word and if $a \in \mathbb{Z}_q$ maximises the function $\mathrm{Re}(\omega^{-a} R)$, then $[aa \ldots a]$ is a maximum likelihood estimate of the transmitted codeword from $RM_q(0, m)$, in the sense of minimising Euclidean distance.

A simple calculation shows that the computational requirement of Algorithm 2 is roughly $(m+2)2^{m-1}$ complex additions and $(m+2)2^{m-1}$ complex multiplications. The equivalent number of real operations (regarding one complex multiplication as requiring 3 real multiplications and 5 real additions and one complex addition as requiring 2 real additions) is $7(m+2)2^{m-1}$ real additions and $3(m+2)2^{m-1}$ real multiplications. Step 4 can be performed using no complex multiplications when $q = 2$ or $q = 4$ by manipulating real and imaginary parts of the r_i. This reduces the computational burden.

Algorithm 2 is a signal-domain, soft-decision decoder, but it can be used to obtain a coding-domain, soft decision decoder too, simply by mapping a real-valued received vector r into its complex analogue with components ω^{r_i}. This coding-domain decoder uses less arithmetic operations than the soft-decision version of Algorithm 1, although they are complex rather than real operations.

4 Majority Logic Decoding Algorithms for Second-order Codes

In this section we consider majority logic-based decoding algorithms for the codes $RM_q(2, m)$ and $ZRM_q(2, m)$, and for codes formed from a union of cosets of $RM_q(1, m)$ contained in either of these codes.

For fixed j and k with $0 \le j < k < m$, we define $\mathcal{I}_{jk}$ to be the set of integers i with $0 \le i = \sum_{\alpha=0}^{m-1} i_\alpha 2^\alpha < 2^m$ with the property that $i_j = i_k = 0$. Given f, a length 2^m vector over $\mathbb{Z}_q$, we define f^{jk} to be the length 2^{m-2} vector with components f_i^{jk}, $i \in \mathcal{I}_{jk}$ where

$$f_i^{jk} = f_{i+2^j+2^k} - f_{i+2^j} - f_{i+2^k} + f_i \bmod q, \quad i \in \mathcal{I}_{jk}$$

Recall that a typical codeword f of such a code is identified with a generalised Boolean function of order 2 with coefficients from $\mathbb{Z}_q$. We write:

$$f = a + \sum_{k=0}^{m-1} a_k x_k + \sum_{0 \le j < k < m} q_{jk} x_j x_k.$$

Repeated application of Lemma 2 shows that in this case f^{jk} is the length 2^{m-2} constant vector $[q_{jk} q_{jk} \ldots q_{jk}]$. This observation forms the basis for our decoding algorithms for second-order codes.

4.1 Majority Logic Decoders for $\mathbf{RM_q(2,m)}$ and $\mathbf{ZRM_q(2,m)}$

We begin with a decoder for the code $RM_q(2,m)$.

Algorithm 3 *(Hard-decision coding-domain decoding algorithm for $RM_q(2,m)$).*

1. *Input the received word $r = [r_0 r_1 \dots r_{2^m-1}]$ as a vector over $\mathbb{Z}_q$.*
2. *For $0 \le j < k < m$, compute the length 2^{m-2} vector r^{jk}.*
3. *For $0 \le j < k < m$, find the most frequent symbol in r^{jk}. Denote this symbol by $\hat{q}_{jk}$.*
4. *Let $r' = r - \sum_{0 \le j < k < m} \hat{q}_{jk} x_j x_k \bmod q$.*
5. *Pass r' to the decoder of Algorithm 1, which outputs integers $\hat{a}, \hat{a}_0, \hat{a}_1, \dots, \hat{a}_{m-1}$.*
6. *Output $\hat{q}_{jk}$, $0 \le j < k < m$.*

Algorithm 3 obtains estimates $\hat{q}_{jk}$ for the second-order coefficients of the transmitted codeword (steps 2 and 3), re-encodes using these estimates to obtain the second-order part $\sum_{0 \le j < k < m} q_{jk} x_j x_k$ and subtracts this word from r (step 4) to reduce the decoding problem to a first-order one (step 5). The algorithm is identical to the Reed majority logic algorithm for the code $RM(2,m)$ when $q = 2$. We have an analogue of Theorem 3, with a very similar proof:

Theorem 4 *Algorithm 3 is a Hamming and Lee distance hard-decision decoder for the code $RM_q(2,m)$. In other words, the algorithm correctly decodes the received vector r in the presence of a error e whose Hamming or Lee weight is at most 2^{m-3}.*

The computational cost of the algorithm is roughly $\binom{m}{2} \cdot 2^m$ arithmetic operations modulo q (steps 2 and 3) plus the cost of a first-order decoding (step 5). It can be extended to soft-decision decoding in the same way as Algorithm 1. It can then also be used to decode the code $ZRM_q(2,m)$, simply by restricting the coefficients $\hat{q}_{jk}$ in the soft version of step 3 to be chosen only from the set of even integers in $\mathbb{Z}_q$. The resulting soft-decision decoder for $ZRM_q(2,m)$ can correct all errors of soft Lee weight less than 2^{m-2}.

We also have a signal-domain version of the above algorithm:

Algorithm 4 *(Soft-decision signal-domain decoding algorithm for $RM_q(2,m)$).*

1. *Input the received word $r = [r_0 r_1 \dots r_{2^m-1}]$ as a vector over $\mathbb{C}$.*
2. *For $0 \le j < k < m$, compute the complex number*

$$R_{jk} = \sum_{i \in \mathcal{I}_{jk}} r_{i+2^j+2^k} \cdot r^*_{i+2^j} \cdot r^*_{i+2^k} \cdot r_i.$$

3. *For $0 \le j < k < m$, select from $\mathbb{Z}_q$ the value $\hat{q}_{jk}$ which maximises $Re(\omega^{-\hat{q}_{jk}} R_{jk})$.*
4. *Compute the codeword $z = \sum_{0 \le j < k < m} \hat{q}_{jk} x_j x_k \bmod q$ and the length 2^m complex vector r' with components $r'_i = r_i \cdot \omega^{-z_i}$.*
5. *Pass r' to the decoder of Algorithm 2, which outputs integers $\hat{a}, \hat{a}_0, \hat{a}_1, \dots, \hat{a}_{m-1}$.*
6. *Output $\hat{q}_{jk}$, $0 \le j < k < m$.*

The above algorithm is readily modified to provide a decoder for $ZRM_q(2,m)$ by restricting $\hat{q}_{jk}$ in Step 3 to be chosen from the set of even integers in $\mathbb{Z}_q$. The computational requirements of the algorithm are dominated by steps 2 and 4, which require a total of roughly $(3\binom{m}{2} + 4)2^{m-2}$ complex multiplications and $\binom{m}{2}2^{m-2}$ complex additions. The equivalent numbers of real operations are $(9\binom{m}{2} + 12)2^{m-2}$ real multiplications and $(17\binom{m}{2} + 20)2^{m-2}$ real additions.

Finally in this section, we note that a general procedure for reducing the decoding of $RM_q(r,m)$ to that of $RM_q(r-1,m)$ can be obtained by generalising the above algorithms. In the coding-domain version, we compute $\binom{m}{r}$ vectors $r^{j_0 j_1 \cdots j_{r-1}}$ of length 2^{m-r}, where each component i of $r^{j_0 j_1 \cdots j_{r-1}}$ is formed from a sum of 2^r terms:

$$r_i^{j_0 j_1 \cdots j_{r-1}} = \sum_{[j_0 j_1 \cdots j_{r-1}] \in \{0,1\}^r} (-1)^{1+wt_H[j_0 j_1 \cdots j_{r-1}]} \cdot r_{i+2^{j_0}+2^{j_1}+\cdots 2^{j_{r-1}}} \bmod q, \quad i \in \mathcal{I}_{j_0 j_1 \cdots j_{r-1}}.$$

Here $\mathcal{I}_{j_0 j_1 \cdots j_{r-1}}$ denotes the set of integers i with $0 \le i = \sum_{\alpha=0}^{m-1} i_\alpha 2^\alpha < 2^m$ with the property that $i_{j_0} = i_{j_1} = \cdots = i_{j_{r-1}} = 0$.

For hard-decision decoding, the coefficient of the monomial $x_{j_0} x_{j_1} \ldots x_{j_{r-1}}$ in the transmitted codeword c is determined from the most common symbol in the vector $r^{j_0 j_1 \cdots j_{r-1}}$. The decoding problem can then be reduced to decoding an order $r-1$ code by re-encoding the order r part of c and subtracting it from r. Soft-decision and signal-domain decoding algorithms can also be derived.

4.2 Majority Logic Decoders for Coset Codes

In this subsection, we present a decoding algorithm that is applicable to a code $\mathcal{C}$ formed from a union of cosets of $RM_q(1,m)$ inside either $RM_q(2,m)$ or $ZRM_q(2,m)$. Let

$$\mathcal{C} = \bigcup_{0 \le i < \ell} \{Q^i + RM_q(1,m)\}$$

where

$$Q^i = \sum_{0 \le j < k < m} q_{jk}^i x_i x_j \bmod q, \quad 0 \le i < \ell$$

are the coset representatives in $\mathcal{C}$.

The basic idea of our algorithm is to assign a likelihood measure to each possible value of each second-order coefficient $\hat{q}_{kl}$ (as in Algorithm 3) by calculating $d_L(r^{jk}, [aa \ldots a])$ for each $a \in \mathbb{Z}_q$. We then calculate a likelihood measure for each coset representative Q^i in turn as a sum of Lee distances:

$$d_i = \sum_{0 \le j < k < m} d_L(r^{jk}, [q_{jk}^i q_{jk}^i \ldots q_{jk}^i]).$$

We choose for our coset representative one which minimises this sum. We then subtract this coset representative from the received codeword and use a soft-decision first-order decoder.

The details of the coding-domain algorithm are as follows

Algorithm 5 *(Coding-domain decoding algorithm for coset code).*

1. *Input the received word $r = [r_0 r_1 \ldots r_{2^m-1}]$ as a vector of real numbers.*

2. *For $0 \le j < k < m$, compute the length 2^{m-2} vector r^{jk}.*

3. *For $0 \le j < k < m$ and $a \in \mathbb{Z}_q$, calculate $d_L(r^{jk}, [aa \ldots a])$.*

4. *For each $0 \le i < \ell$, calculate (using the results from step 3):*

$$d_i = \sum_{0 \le j < k < m} d_L(r^{jk}, [q^i_{jk} q^i_{jk} \ldots q^i_{jk}]).$$

5. *Select the value I which minimises d_I, $0 \le I < \ell$.*

6. *Let $r' = r - \sum_{0 \le j < k < m} q^I_{jk} x_j x_k \bmod q$.*

7. *Pass r' to the soft-decision version of Algorithm 1, which outputs integers $\hat{a}, \hat{a}_0, \hat{a}_1, \ldots, \hat{a}_{m-1}$.*

8. *Output I.*

Despite the apparently ad hoc nature of our likelihood measures, we are able to show that the above algorithm can correct all error patterns of weight less than half the minimum distance of the code $RM_q(2, m)$:

Theorem 5 *Let $\mathcal{C}$ be a union of cosets of $RM_q(1, m)$ inside $RM_q(2, m)$. Then Algorithm 5 correctly decodes $\mathcal{C}$ in the presence of any error of soft Lee weight less than 2^{m-3}.*

Proof: Let $c = a + \sum_{i=0}^{m-1} a_k x_k + Q^I$ be the transmitted codeword, and let r be the received word with components $r_i \in [0, q)$. Write $e = r - c \bmod q$ and suppose $wt_L(e) < 2^{m-3}$. Now, for each j, k with $0 \le j < k < m$, we have $r^{jk} = c^{jk} + e^{jk} \bmod q$. But c^{jk} is equal to the vector $[q^I_{jk} q^I_{jk} \ldots q^I_{jk}]$. Since e has soft Lee weight less than 2^{m-3}, so does the vector e^k. Hence

$$d_L(r^{jk}, [q^I_{jk} q^I_{jk} \ldots q^I_{jk}]) = wt_L(e^k) < 2^{m-3}. \tag{2}$$

Also, for any $a \in \mathbb{Z}_q$ with $a \ne q^I_{jk}$, we have (using a triangle inequality for the Lee metric and the fact that the minimum Lee distance of the code $RM_q(0, m-2)$ is exactly 2^{m-2}):

$$\begin{aligned} d_L(r^{jk}, [aa \ldots a]) &\ge d_L([q^I_{jk} q^I_{jk} \ldots q^I_{jk}], [aa \ldots a]) - d_L(r^{jk}, [q^I_{jk} q^I_{jk} \ldots q^I_{jk}]) \\ &> 2^{m-2} - 2^{m-3} \\ &= 2^{m-3} \end{aligned} \tag{3}$$

Thus for any i with $0 \le i \ne I < \ell$, we have:

$$\begin{aligned} d_i - d_I &= \sum_{0 \le j < k < m} d_L(r^{jk}, [q^i_{jk} q^i_{jk} \ldots q^i_{jk}]) - d_L(r^{jk}, [q^I_{jk} q^I_{jk} \ldots q^I_{jk}]) \\ &= \sum_{0 \le j < k < m,\ q^i_{jk} \ne q^I_{jk}} d_L(r^{jk}, [q^I_{jk} q^I_{jk} \ldots q^I_{jk}]) - d_L(r^{jk}, [q^i_{jk} q^i_{jk} \ldots q^i_{jk}]) \end{aligned}$$

The sum here is non-empty because the Q^i are distinct coset representatives. Moreover, from inequalities (2) and (3), each term of the sum is positive. Thus $d_i > d_I$ and so choosing the value of i which minimises d_i in step 5 correctly identifies the coset representative Q^I.

Finally, because the soft-decision version of Algorithm 1 can correct errors e of soft Lee weight less than 2^{m-2}, steps 6 and 7 correctly decode the first-order part of the codeword c. □

Algorithm 5 can also be adapted to decode an arbitrary union of cosets of $RM_q(1, m)$ inside $ZRM_q(2, m)$. This latter code has minimum Lee distance double that of $RM_q(2, m)$. We need only replace step 3 of the algorithm with a more efficient step:

3′. For $0 \le j < k < m$ and $a \in 2\mathbb{Z}_q$, calculate $d_L(r^{jk}, [aa \dots a])$.

The rest of the algorithm is identical. The following theorem shows that this adapted algorithm does take full advantage of the increased minimum distance.

Theorem 6 *Let $\mathcal{C}$ be a union of cosets of $RM_q(1,m)$ inside $ZRM_q(2,m)$. Then Algorithm 5 correctly decodes $\mathcal{C}$ in the presence of any error of soft Lee weight less than 2^{m-2}.*

Proof: The proof is very similar to that of Theorem 5. Given an error vector e with $wt_L(e) < 2^{m-2}$, we can derive inequalities:

$$d_L(r^{jk}, [q^I_{jk} q^I_{jk} \dots q^I_{jk}]) < 2^{m-2}.$$

and

$$d_L(r^{jk}, [aa \dots a]) > 2^{m-2} \quad \text{for all } a \in 2\mathbb{Z}_q, a \neq q^I_{jk},$$

where the latter inequality holds because the minimum Lee distance of the code $ZRM_q(0, m-2)$ is equal to 2^{m-1}. The proof now follows the pattern set in the proof of Theorem 5. □

The computational requirement of steps 1 – 5 of the above algorithm is roughly $(q+2)\binom{m}{2}2^{m-1} + \ell\binom{m}{2}$ real additions. The first term here is a fixed overhead, roughly equivalent to the cost of the soft-decision version of Algorithm 3. This term can be roughly halved when step 3′ is used for decoding a coset code in $ZRM_q(2,m)$. It can be further reduced by computing the distances $d_L(r^{jk}, a)$ over only the set of values $a \in A_{jk} = \{q^i_{jk} : 0 \le j < k < m\}$ which actually occur as coefficients q^i_{jk} in some coset representative Q^i in the list of cosets. The second term $\ell\binom{m}{2}$ is, for values of m that are of interest, a small multiple of the number of cosets ℓ in the code. Thus the algorithms are highly efficient, even for codes formed from large numbers of cosets.

We have presented Algorithm 5 as a soft-decision algorithm. Of course it can also be used for hard-decision decoding. This allows the use of only integer operations and leads to a simplified implementation. There is also a natural signal-domain version of the algorithm:

Algorithm 6 *(Signal-domain decoding algorithm for coset code).*

1. *Input the received word $r = [r_0 r_1 \dots r_{2^m-1}]$ as a vector of complex numbers.*
2. *For $0 \le j < k < m$, compute the complex number*

$$R_{jk} = \sum_{i \in \mathcal{I}_{jk}} r_{i+2^j+2^k} \cdot r^*_{i+2^j} \cdot r^*_{i+2^k} \cdot r_i.$$

3. *For $0 \le i < \ell$, compute the real number d_i, where:*

$$d_i = Re\left(\sum_{0 \le j < k < m} \omega^{-q^i_{jk}} R_{jk} \right),$$

4. *Select the value I which maximises d_I, $0 \le I < \ell$.*
5. *Compute the codeword $z = \sum_{0 \le j < k < m} q^I_{jk} x_j x_k \bmod q$ and the length 2^m complex vector r' with components $r'_i = r_i \cdot \omega^{-z_i}$.*

6. *Pass r' to the decoder of Algorithm 2, which outputs integers $\hat{a}, \hat{a}_0, \hat{a}_1, \ldots, \hat{a}_{m-1}$.*

7. *Output I.*

The computational requirement of steps 1 – 3 of the above algorithm is roughly $3\binom{m}{2}2^{m-2} + \ell\binom{m}{2}$ complex multiplications and $\binom{m}{2}2^{m-2}+\ell\binom{m}{2}$ complex additions. The equivalent numbers of real operations are $9\binom{m}{2}2^{m-2}+3\ell\binom{m}{2}$ real multiplications and $17\binom{m}{2}2^{m-2}+7\ell\binom{m}{2}$ real additions.

Notice that when $q = 2$ or $q = 4$, step 3 can be carried out using only complex additions, sign changes and manipulation of real and imaginary parts of the R_{jk}. This reduces the number of complex multiplications required by $\ell\binom{m}{2}$.

5 Reduction Decoders

Recall from Lemma 2 that if $f = \sum_{0\le j<k<m} q_{jk}x_jx_k + \sum_{k=0}^{m-1} a_kx_k + a$ is a word of $RM_q(2,m)$, then for each k with $0 \le k < m$, the word f^k is a word of $RM_q(1, m-1)$. In fact, we have

$$f^k = \sum_{j\neq k} q_{jk}x_j + a_k,$$

a function in variables $x_0, \ldots, x_{k-1}, x_{k+1}, \ldots, x_{m-1}$ obtained by computing

$$f(x_0, ..., x_{k-1}, 1, x_{k+1}, \ldots, x_{m-1}) - f(x_0, ..., x_{k-1}, 0, x_{k+1}, \ldots, x_{m-1}) \bmod q.$$

Thus all the first and second-order coefficients q_{jk} and a_k arising in f can be obtained from the m words f^k.

This observation forms the basis of a second set of decoding algorithms. We see that if r is a received vector, then after computing the m vectors r^k, the decoding problem for $RM_q(2,m)$ is reduced to m first-order decoding problems for $RM_q(1, m-1)$. By appropriately re-encoding using the coefficients obtained from these m first-order results, the constant coefficient can also be estimated. Any first-order decoder we like can be employed for the m sub-problems (for example, the majority logic algorithms developed above or the FHT). The idea is easily extended to show that the decoding problem for the code $RM_q(r,m)$ can be reduced to m decodings of $RM_q(r-1, m-1)$ and estimation of a constant coefficient.

In what follows, we will present a variety of 'reduction decoders' both for the full second-order code and for codes formed from cosets of $RM_q(1,m)$ inside $RM_q(2,m)$ and $ZRM_q(2,m)$.

5.1 Second-order Reduction Decoders

We begin by describing a hard-decision decoder for $RM_q(2,m)$. Assume that we have available a hard-decision, coding-domain decoder for $RM_q(1, m-1)$. We assume that its input is a length 2^{m-1} vector over $\mathbb{Z}_q$ and that its output is a list of coefficients $\hat{a}, \hat{a}_0, \ldots, \hat{a}_{m-2}$ giving estimates for the constant and first-order terms of a codeword of $RM_q(1, m-1)$. See Algorithm 1 for an example of such an algorithm.

Algorithm 7 *(Hard-decision coding-domain decoding algorithm for $RM_q(2,m)$).*

1. *Input the received word $r = [r_0r_1 \ldots r_{2^m-1}]$ as a vector over $\mathbb{Z}_q$.*

2. *For $0 \le k < m$, compute the length 2^{m-1} vector r^k.*

3. *For $0 \leq k < m$, denote the output of the decoder for $RM_q(1, m-1)$ on input r^k by*

$$\hat{a}_k, \hat{q}_{0k}, \ldots, \hat{q}_{(k-1)k}, \hat{q}_{(k+1)k} \cdots, \hat{q}_{(m-1)k}.$$

4. *Let $r' = r - \sum_{0 \leq j < k < m} \hat{q}_{jk} x_j x_k - \sum_{k=0}^{m-1} \hat{a}_k x_k \bmod q$.*

5. *Let $\hat{a}$ be the most frequent symbol in r'.*

6. *Output integers $\hat{a}, \hat{a}_0, \hat{a}_1, \ldots, \hat{a}_{m-1}$, $\hat{q}_{jk}$, $0 \leq j < k < m$.*

A number of remarks on this algorithm are in order.

Because of an inherent symmetry, the algorithm actually provides two separate estimates for each second-order coefficient q_{jk} (where $j < k$), these being denoted by $\hat{q}_{jk}$ and $\hat{q}_{kj}$ in step 3. Step 6 outputs one of the two. It is not hard to see that if the decoder of Algorithm 1 is used in step 3, then the two estimates will be identical, and moreover will be the same as that provided by Algorithm 3. This suggests that Algorithm 7 is performing twice as much computation as is necessary, and this is reflected in the fact that its complexity is roughly twice that of Algorithm 3.

It is straightforward to prove, using similar ideas as were used to prove Theorem 3, that the algorithm is a Hamming and Lee distance decoder for the code $RM_q(2, m)$.

Soft-decision and signal-domain versions of the algorithm are readily developed. The hard-decision coding-domain decoder for $RM_q(1, m-1)$ should be replaced by an appropriate algorithm — for example, either the soft-decision version of Algorithm 1 or Algorithm 2. In the signal-domain case, the generalised FHT of [4] can be used to perform the m decoding steps for $RM_q(1, m-1)$. We will show below how the extra information available in the output vector of the generalised FHT can be exploited to develop efficient decoders for codes formed from unions of second-order cosets of $RM_q(1, m)$.

5.2 Reduction Decoder for Coset Codes

Now we describe a signal-domain coset decoder. The main idea is to use Algorithm 7 with a generalised FHT as the first order decoder, but to use information obtained from the FHT output vectors to judge each coset representative in turn. This approach is similar in spirit to that taken in Algorithm 6.

As in Section 4.2, let $\mathcal{C}$ be the code $\bigcup_{0 \leq i < \ell} \{Q^i + RM_q(1, m)\}$ where

$$Q^i = \sum_{0 \leq j < k < m} q^i_{jk} x_i x_j \bmod q, \quad 0 \leq i < \ell$$

are the coset representatives in $\mathcal{C}$. For $0 \leq i < \ell$ and $0 \leq k < m$, let v^{ik} denote the vector

$$[q^i_{0k}, q^i_{1k}, \ldots, q^i_{(k-1)k}, q^i_{k(k+1)}, \ldots, q^i_{k(m-1)}]$$

obtained by extracting the coefficients of second-order terms involving x_k from the quadratic form Q^i. These vectors will be used to index entries in vectors output by generalised FHTs in our algorithm. They can be computed from lists of coefficients q^i_{jk} on the fly, or can be computed ahead of time and held in storage.

Algorithm 8 *(Signal-domain reduction decoding algorithm for coset code).*

1. *Input the received word* $r = [r_0 r_1 \dots r_{2^m-1}]$ *as a vector of complex numbers.*

2. *For* $0 \le k < m$, *compute the length* 2^{m-1} *vector* r^k *where*

$$r_i^k = r_{i+2^k} \cdot {r_i}^*, \quad i \in \mathcal{I}_k.$$

3. *For* $0 \le k < m$, *compute the* q*-ary generalised FHT of the vector* r^k, *denoted by* Y^k, *as defined in equation (1).* (Y^k *is a length* q^{m-1} *vector with complex components labelled* $Y^k_{[a_0,a_1,\dots,a_{m-2}]}$).

4. *For each* $0 \le i < \ell$, *calculate (using the results from step 3) the following sum of squared absolute values of entries from the vectors* Y^k*:*

$$d_i = \sum_{0 \le k < m} |Y^k_{v^{ik}}|^2.$$

5. *Select the value* I *which maximises* d_I, $0 \le I < \ell$.

6. *For* $0 \le k < m$, *let* $\hat{a}_k$ *denote the integer in* $\mathbb{Z}_q$ *which maximises* $Re(\omega^{-\hat{a}_k} Y^k_{v^{Ik}})$.

7. *Compute the codeword* $z = \sum_{0 \le j < k < m} q^I_{jk} x_j x_k + \sum_{k=0}^{q-1} \hat{a}_k x_k \bmod q$ *and the complex number*

$$R' = \sum_{i=0}^{2^m-1} r_i \cdot \omega^{-z_i}.$$

8. *Select from* $\mathbb{Z}_q$ *the value* $\hat{a}$ *which maximises* $Re(\omega^{-\hat{a}} R')$.

9. *Output* $\hat{a}, \hat{a}_0, \hat{a}_1, \dots, \hat{a}_{m-1}$ *and* I.

The main computational burden of the algorithm divides naturally into two parts. The first part is a fixed overhead associated with steps 2 and 3. Here the m vectors r^k are calculated (requiring a total of $m2^{m-1}$ complex multiplications) and then their q-ary FHTs are computed. For $q = 2$, the cost of each FHT is approximately $(m-1)2^{m-1}$ complex additions, while for $q = 4$, the cost is $2^{2(m-1)}$ complex additions. For $q = 8$, the cost per FHT is around $2^{3(m-1)}$ complex multiplications and $2^{3(m-1)}$ complex additions. The computations in the second part, step 4, can be carried out using around $2m$ real additions and $2m$ real multiplications per coset.

A very important optimisation can be made when the code $\mathcal{C}$ consists of cosets of $RM_q(1,m)$ contained in $ZRM_q(2,m)$ (rather than in $RM_q(2,m)$), where can replace the q-ary FHTs in step 3 by $(q/2)$-ary FHTs. The key point is that the cost of a $(q/2)$-ary FHT is much lower than that of a q-ary one. The reason this replacement step can be used in this situation is that the words r^k can be regarded as noisy versions of the words in $ZRM_q(1,m-1)$ instead of $RM_q(1,m-1)$. Maximum likelihood decoding of the code $ZRM_q(1,m-1)$ can be achieved by applying a $(q/2)$-ary FHT to the complex received vector, using the component of maximum absolute value in Y, say $Y_{[\hat{a}_0\hat{a}_1\dots\hat{a}_{m-2}]}$, to determine estimates $2\hat{a}_0, 2\hat{a}_1, \dots, 2\hat{a}_{m-2}$ for the $m-1$ first order coefficients and then estimating $\hat{a}$, the q-ary constant term, to be the integer in $\mathbb{Z}_q$ which maximises $\mathrm{Re}(\omega^{-\hat{a}} Y_{[\hat{a}_0\hat{a}_1\dots\hat{a}_{m-2}]})$. In the optimised version of Algorithm 8, we re-define v^{ik} to be the vector

$$\frac{1}{2}[q^i_{0k}, q^i_{1k}, \dots, q^i_{(k-1)k}, q^i_{k(k+1)}, \dots, q^i_{k(m-1)}]$$

and have a replacement step 3:

3′. For $0 \leq k < m$, compute the $(q/2)$-ary generalised FHT of the vector r^k, denoted by Y^k, as defined in equation (1). (Y^k is now a length $(q/2)^{m-1}$ vector with complex components labelled $Y^k_{[a_0,a_1,\ldots,a_{m-2}]}$, where $a_t \in \mathbb{Z}_{q/2}$).

Further optimisation is possible if all the second-order coefficients of codewords are known to lie in a set $q'\mathbb{Z}_q$ where q' divides q. For then the q-ary FHTs can be replaced by (q/q')-ary FHTs. This observation is particularly useful for a particular class of 2^h-ary codes having PMEPR at most 2 that were identified in [2, Corollary 3.4]: in this case we can take $q' = 2^{h-1}$ and all the 2^h-ary FHTs become binary FHTs with complex inputs.

For a code formed from a union of ℓ cosets of $RM_q(1,m)$ in $ZRM_q(2,m)$ (like that in example 2), the computations in the two groups identified above require $(2m+3)m2^{m-1} + 2m\ell$ real additions and $3m2^{m-1} + 2m\ell$ real multiplications for $q = 4$ and $m2^{2m-1} + 5m2^{m-1} + 2m\ell$ real additions and $3m2^{m-1} + 2m\ell$ real multiplications for $q = 8$.

6 Complexity Comparisons and Simulation Results

6.1 Complexity Comparisons

We wish to compare the computational complexities of our Algorithms 6 and 8 for coset codes with the complexities of existing approaches to decoding such codes summarised in Section 2.3. We use as our basic measure the number of real arithmetic operations (additions and multiplications) required to carry out each algorithm. In our complexity estimates for Algorithms 6 and 8, we have already ignored small numbers of operations, concentrating only on the main algorithmic steps. For small numbers of cosets and short codes, these operations may account for a significant fraction of the total cost. The same is true for the algorithms in Section 2.3.

Both of our algorithms have a fixed overhead associated with manipulations of the received vector (and possibly computing FHTs) and then a second cost which scales linearly with the number ℓ of cosets in the code. For Algorithm 6, this per coset cost is $10\binom{m}{2}$ operations, while for Algorithm 8 it is $4m$ operations. In both cases, the per coset cost is small. This is in contrast to the maximum likelihood approach in [4, Algorithm 1] and the more efficient version of [2, Algorithm 5.3]: here there is no fixed overhead, but the per coset cost can be quite high (on the order of $m2^m$ arithmetic operations for the latter algorithm, potentially much higher for the former). Thus we should expect our algorithms to be more efficient when the number of cosets is large. On the other hand, it is worth pointing out that [2, Algorithm 5.3] requires no multiplication operations, only additions, and that these are far simpler to implement in hardware or on a DSP.

For concreteness, in Table 1 we compare the decoding complexities of Algorithms 6 and 8 with the algorithms from [2, 4] for two particular choices of code. The first code is the code of Example 2, the second consists of 256 cosets of $RM_4(1,4)$ in $RM_4(2,4)$. The table illustrates the decrease in decoding complexity offered by our new algorithms, particularly over that of maximum likelihood decoding as presented in [4]. Even though the per coset cost of Algorithm 8 is lower than that of Algorithm 6, the former algorithm is more expensive to implement for these particular codes because of the relative sizes of fixed overheads.

6.2 Simulation Results

We have simulated the maximum likelihood algorithm [4, Algorithm 1], [2, Algorithm 5.3], the soft version of the quaternary algorithm of [5] and Algorithms 6 and 8 with the code of Example

Number of cosets	Operation	[4, Algorithm 1]	[2, Algorithm 5.3]	Algorithm 6	Algorithm 8
32	+	30720	2112	792	608
	×	0	0	216	352
256	+	245760	16448	3480	2400
	×	0	0	216	2144

Table 1: Computational requirements of decoders for codes formed from cosets of $RM_4(1,4)$ in $RM_4(2,4)$ (equivalent real additions and multiplications)

2 for two different channel models. Recall that this code has rate 0.47, minimum Hamming distance at least 4 and minimum Lee distance at least 8, and a PMEPR of 6.0dB.

Firstly, we describe the discrete Additive White Gaussian Noise (AWGN) channel. Let F^{-1} denote the IFFT of the OFDM modulator, and let $y = F^{-1}(\omega^c)$ where $c = (c_0, c_1, \ldots, c_{2^m-1})$ is the OFDM codeword. So y is a sampled version of the continuous time-domain OFDM signal. Then we can write:

$$r = F(y+n) = F(y) + F(n) = \omega^c + F(n)$$

where r denotes the demodulated complex-valued vector that is input to a decoder and

$$n = (n_0, n_1, \ldots, n_{2^m-1})$$

is a vector whose components are i.i.d. complex-valued AWGN samples with zero mean and variance $\sigma^2 = N_0/2$. Here N_0 denotes the one-sided noise spectral density. At the receiver, we use coherent detection and assume perfect synchronisation. For validation purposes we first considered the BER performance for the uncoded case, which has a theoretical bit error probability given by [10, equation (5-2-57)]:

$$P = \frac{1}{2}\text{erfc}\left(\sqrt{\frac{E_b}{N_0}}\right) \tag{4}$$

where erfc(.) is the complementary error function, $E_b = C/T_b$, C is the average carrier power, and T_b is the bit period of the information in seconds. The simulated results for the uncoded case are shown in Figure 1 as the points marked $*$. Also included in the figure is the continuous curve given by evaluating (4) over the range of E_b/N_0. It is clear that the uncoded BER performance is in good agreement with the theoretical curve.

The simulation results for the various decoders on the AWGN channel are also shown in Figure 1. We see that, with the exception of Algorithm 6, all the decoders perform within a few tenths of a dB of maximum likelihood decoding, and so there is little to choose between them in performance terms. But, as we have seen above, the algorithms have quite different implementation complexities, with Algorithm 8 having the lowest complexity for this code. We note that, despite the fact that the coding-domain version of Algorithm 6 was proved to be a Lee and Hamming distance decoder (see Theorem 5), the signal-domain version performs rather badly. For this reason we caution against over-reliance on such distance measures. We also expect the other majority logic based decoders to have poor performance. Finally, we note the perhaps surprising fact that the best performing algorithm in Gaussian noise is the soft version of the 'Gray map' decoder of [5]. The underlying reason that this algorithm out-performs the MLSD decoder (in terms of bit-error rate at a given E_b/N_0) may be the additional level of

protection afforded by the use of Gray coding of bits for this particular simulation. (We did not use Gray coding when simulating the other decoding algorithms).

Although the discrete AWGN channel gives a indication of the performance attainable with the various decoding algorithms, it is not a channel model appropriate to typical OFDM applications. Consequently, simulations with that model do not reflect the potential gains that can be obtained using a coded OFDM signal. To make a proper assessment of the various decoding algorithms, we need to define a realistic channel model that provides a suitable platform for comparing performances without introducing undue complexity. Here, we model an idealised OFDM system, where the channel is frequency selective across the complete bandwidth of the OFDM signal, but where the subcarriers are independently faded according to a Rayleigh distribution. This provides sufficient diversity for the OFDM signal to exploit without introducing intersymbol interference between adjacent OFDM symbols. Thus the results obtained from this type of channel model can be considered as placing an upper bound on the performance of a coded OFDM system.

As above, $y = F^{-1}(\omega^c)$ denotes a sampled version of the continuous time-domain OFDM signal. Let $H = (H_0, H_1, ..., H_{2^m-1})$ be a complex vector which represents the frequency domain response of the channel across the bandwidth of the OFDM signal. We write

$$H_j = \alpha_j e^{-i\phi_j}$$

where α_j is a non-negative real number representing the magnitude of the fade on carrier j and ϕ_j denotes the relative phase-offset on carrier j. We assume that the α_j are each independently Rayleigh distributed and that the ϕ_j have independent uniform distributions in $[0, 2\pi)$. Since we are assuming perfect synchronisation, we can in fact take $\phi_j = 0$ for each j. Then we model the effect of the channel on the vector y as:

$$r = \omega^c H + F(n)$$

where r is the demodulated vector input to a decoder and n is Gaussian noise as above, and $\omega^c H$ denotes the vector with components $\omega^{c_j} H_j$, $0 \leq j < 2^m$.

We first consider the effect of the channel in the uncoded case. We simulated 50000 different instantiations of the above channel model with 32 OFDM symbols per instantiation. The theoretical bit error probability for both BPSK and QPSK modulation is given by [10, equation (14-3-7)]:

$$P = \frac{1}{2}\left[1 - \sqrt{\frac{\bar{\gamma}_b}{1+\bar{\gamma}_b}}\right] \tag{5}$$

where $\bar{\gamma}_b = \frac{E_b}{N_0} E((\alpha_j)^2)$ is the average signal to noise ratio across the bandwidth of the OFDM signal and $E((\alpha_j)^2)$ is the expectation of the fade magnitude. The simulated results for the uncoded case are shown in Figure 2 as the points marked $*$. Also included in the figure is the continuous curve given by evaluating (5) over the range of E_b/N_0. It is again clear that the uncoded BER performance is in good agreement with the theoretical curve.

The simulation results for the various decoders with this fading channel model are also shown in Figure 2. We see that there is a greater spread in the performances of the algorithms than in the Gaussian noise case. The signal-domain Algorithm 8 lies within 2dB of the maximum likelihood algorithm of [4], and has much lower implementation complexity. This algorithm also outperforms [2, Algorithm 5.3] by as much as 3dB and is again less complex to implement. In summary, Algorithm 8 is an attractive alternative to maximum likelihood decoding for this type of fading channel.

In contrast to the AWGN situation, the coding-domain algorithms perform relatively poorly — this is attributable to the fact that extraction of phases of badly faded carriers prior to the operation of a coding-domain algorithm leads to greatly amplified errors. The signal-domain algorithms do not suffer from this deficiency, and indeed they indirectly use the magnitudes of received vector components r_i to provide reliability information. For example, in step 2 of Algorithm 8, components r_i that are small in magnitude will lead to small components in the vectors r^k; these vectors are fed directly to FHTs which compute correlations to first-order codewords and components of small magnitude in the input vector to a FHT will have less influence on the FHT outputs than components of large magnitude.

One further conclusion that can be drawn from these results is that it can be important to perform simulations with a channel model appropriate to the application in mind when selecting a decoding algorithm.

7 Conclusions

We have given generalisations of the Reed majority logic decoding algorithm for the classical Reed-Muller codes that is applicable to their non-binary generalisations. We have also made non-trivial modifications to obtain low complexity, high performance decoding algorithms that are usable in practice for coset codes of the type that are of increasing importance in OFDM. We have given a theoretical justification for the first modification by proving that the resulting algorithm is a Hamming and Lee distance decoder. Our second modification lacks this, but in any case has higher performance than the first.

We have made a comparison of our new algorithms with existing decoders, both in terms of complexity and decoding performance. We have demonstrated that one of our decoders (Algorithm 8) gives close to maximum likelihood performance with substantially reduced complexity.

References

[1] J.H. Conway and N.J.A. Sloane. Soft decoding techniques for codes and lattices, including the Golay code and the Leech lattice. *IEEE Trans. Inform. Theory*, IT-32: 41–50, 1986.

[2] J.A. Davis and J. Jedwab. Peak-to-mean power control in OFDM, Golay complementary sequences and Reed-Muller codes. Technical Report HPL-97-158, Hewlett-Packard Labs., Dec. 1997. Submitted to *IEEE Transactions on Information Theory.*

[3] A.J. Grant and R.D.J. van Nee. Efficient maximum likelihood decoding of peak power limiting codes for OFDM. In *IEEE 48th Vehicular Tech. Conf.*, pages 2081–2084, 1998.

[4] A.J. Grant and R.D.J. van Nee. Efficient maximum-likelihood decoding of q-ary modulated Reed-Muller codes. *IEEE Communications Letters*, 2(5):134–136, May 1998.

[5] A.E. Jones and T.A. Wilkinson. Performance of Reed-Muller codes with OFDM and a maximum-likelihood decoding algorithm. Technical Report HPL-98-88, Hewlett-Packard Labs., April 1998. Submitted to *IEEE Transactions on Communications.*

[6] X. Li and L.J. Cimini, Jr. Effects of clipping and filtering on the performance of OFDM. *IEEE Communications Letters*, 2: 131–133, May 1998.

[7] F.J. MacWilliams and N.J.A. Sloane. *The Theory of Error-Correcting Codes (2nd Edition).* North Holland, Amsterdam, 1986.

[8] H. Ochiai and H. Imai. Block coding scheme based on complementary sequences for multicarrier signals. *IEICE Trans. Fundamentals*, pages 2136–2143, Nov. 1997.

[9] K.G. Paterson. Generalised Reed-Muller codes and power control in OFDM. Technical Report HPL-98-57, Hewlett-Packard Labs., March 1998. Submitted to *IEEE Transactions on Information Theory.*

[10] J.G. Proakis. *Digitial Communications (3rd Edition).* McGraw-Hill, Inc., 1995.

[11] R.D.J. van Nee. OFDM codes for peak-to-average power reduction and error correction. In *IEEE Globecom 1996*, pages 740–744, London, Nov. 1996.

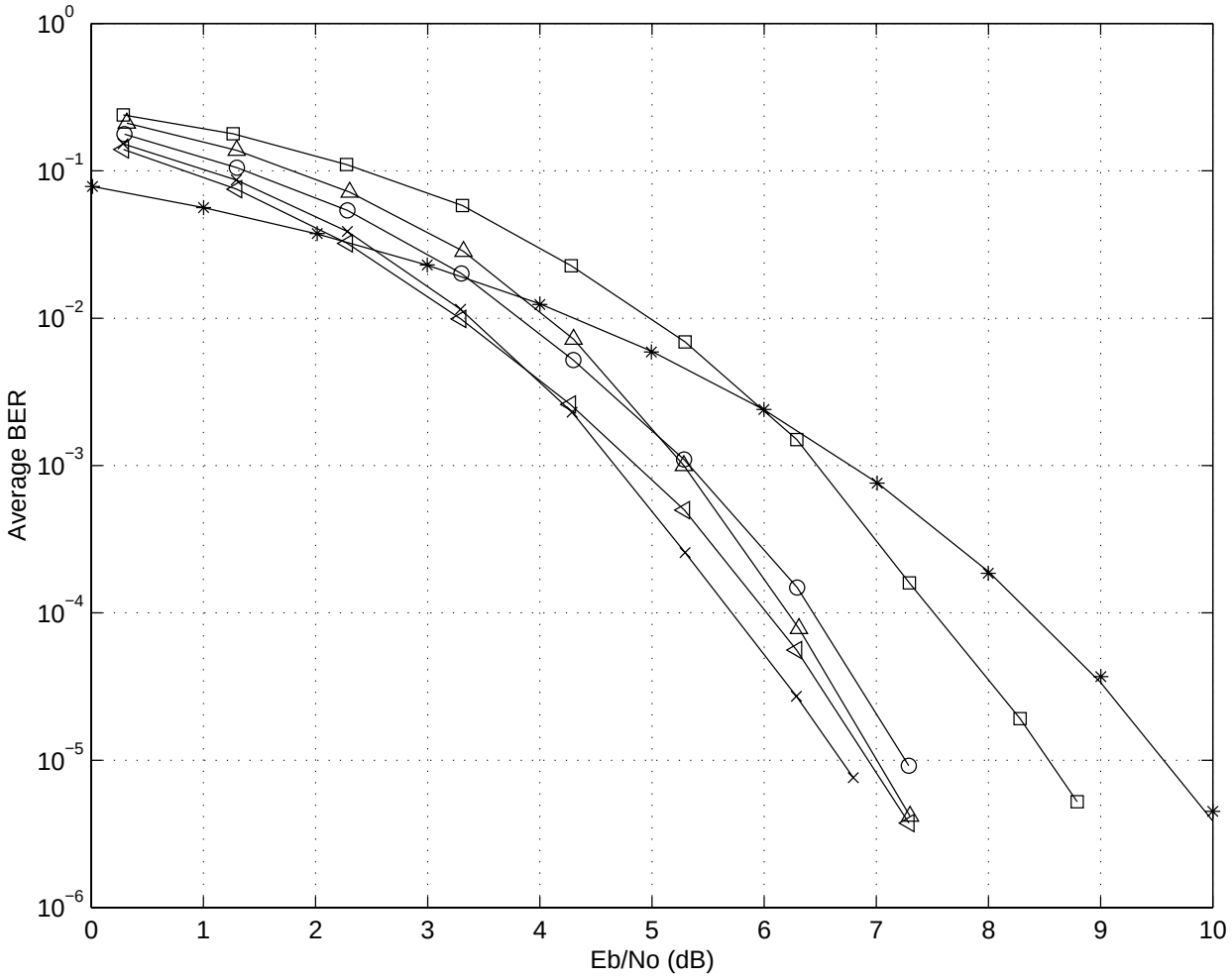

Figure 1: Performance of decoders for AWGN channel.

- $*$ uncoded
- $\lhd$ maximum likelihood decoding, [4, Algorithm 1]
- $\times$ soft Gray mapped decoder from [5]
- $\triangle$ [2, Algorithm 5.3]
- $\square$ Algorithm 6
- $\circ$ Algorithm 8

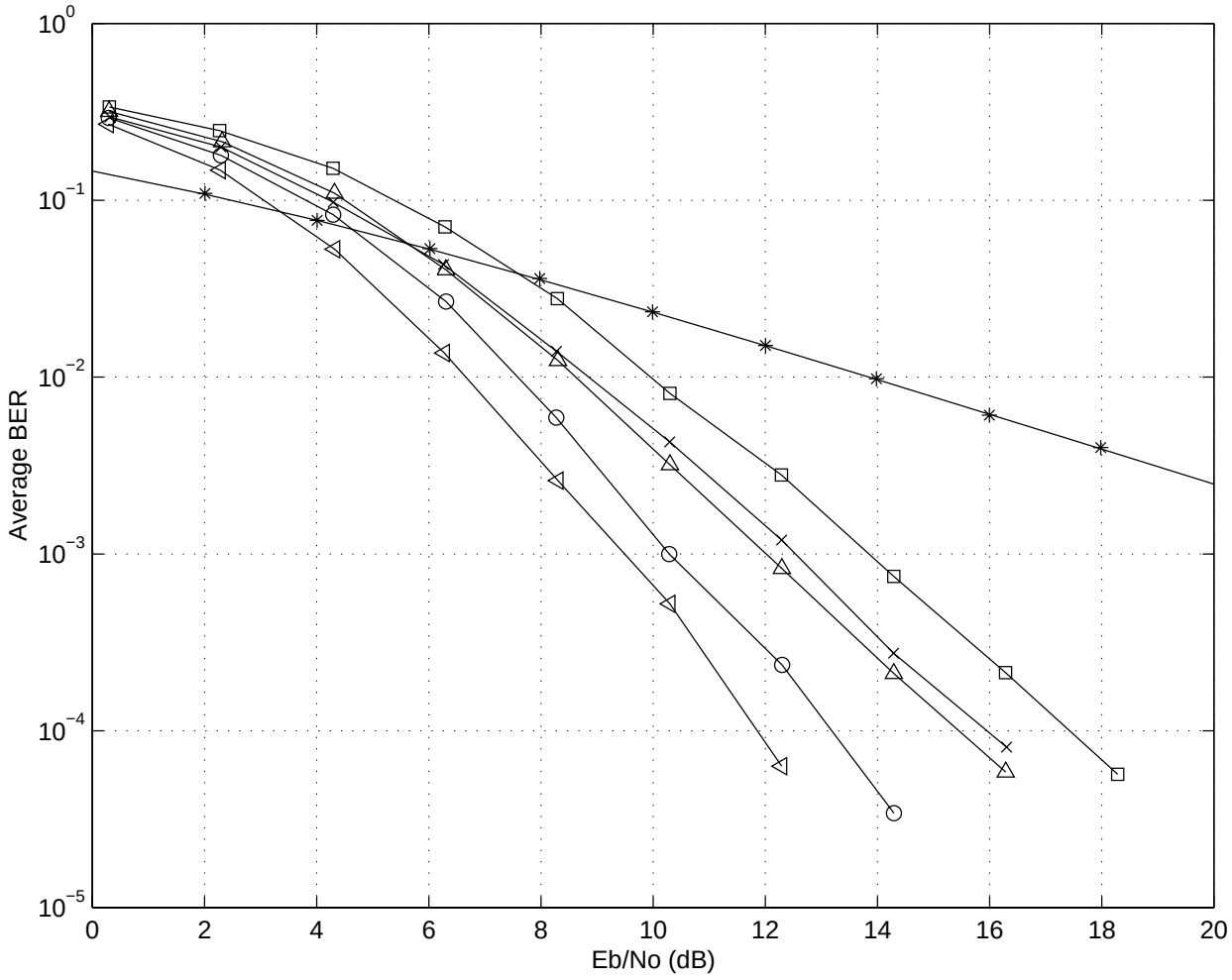

Figure 2: Performance of decoders for multipath fading channel.

- $*$ uncoded
- $\triangleleft$ maximum likelihood decoding, [4, Algorithm 1]
- $\times$ soft Gray mapped decoder from [5]
- $\triangle$ [2, Algorithm 5.3]
- $\square$ Algorithm 6
- $\circ$ Algorithm 8